A GQM-based Approach for Software Process Patterns Recommendation

Zhangyuan Meng†, Cheng Zhang†, Beijun Shen†,Wei Yin‡
†School of Software, Shanghai Jiao Tong University, Shanghai, China
‡China Aeronautical Radio Electronics Research Institute, Shanghai, China
Email: {602389789, jonathenzc, bjshen}@sjtu.edu.cn, yin_wei@careri.com

***Abstract*—A good software process can help project manager manage software development effectively and control development risks. For this reason, theory and experts' experience are concluded and put into process patterns. But it still requires human skills to search for appropriate process patterns in practice. To tackle this challenge, this paper proposes a Goal-Question-Metric (GQM) based approach to recommending software process patterns. The essential idea of this approach is to use a GQM method to design scenario questions for software process patterns, elicit the requirement of new project by answering these questions, and then recommend the optimal matching patterns to the project. In particular, we use a Latent Dirichlet Allocation model on the scenario descriptions of software process patterns to achieve a text-topic distribution, and then apply the K-means method to do text clustering, which facilitate scenario questions design a lot. We evaluate the performance of our topic clustering method by comparing it with that of the statistics method based on TF-IDF. The evaluation results show that our method contributes a high F-score which is 11.6% higher than that of the traditional TF-IDF approach. Furthermore, the average precision of recommendation can reach 57%.**

***Index Terms*—Software Process Pattern Recommendation; Machine Learning; Goal-Question-Metric**

I. Introduction

Nowadays a large number of software systems are under development, meanwhile, software process plays a key role in helping software managers develop high-quality systems and control risks. For aiding software development, process experts have designed many software process patterns [1] [2] [3], including waterfall process model, incremental process model, evolution model, scrum model, test driven development and so on. These process patterns have aggregated considerable experience and knowledge, as they are extracted from many successful projects, refined and improved constantly. Literatures [4] [5] show that these patterns do provide general process solutions that can lead to further successful software projects.

Although many useful process patterns have been designed, it is not easy to find an appropriate one for a given software project. Software development is a knowledge intensive task, and many technical, social, and environment constraints and factors can influence the software development. Choosing an appropriate process pattern is not only an intelligent work, but also a manpower consumption work. The chosen process patterns may not be appropriate if these constraints are not considered. Therefore automated process pattern recommendation with high performance is in urgent need. It can put forward suggestions for the users to choose appropriate process patterns so that the software development can be significantly improved, which means the whole process can be optimized, the redundant development activities can be eliminated, and the quality and maintainability of the software under development can be enhanced.

However, it is non-trivial to recommend software process patterns automatically in practice because of two main difficulties:

1) *How to acquire and express the software process patterns and software project requirements which can be understood by computers?* Software process patterns are summed up by process experts through practical experience. There is no pattern language standard to unify formats and styles of process patterns, which makes them unstructured and difficult for computers to understand and process. There is the same problem with the software project requirements.
2) *How to recommend appropriate software process patterns according to the project requirements automatically?* Traditionally, software process patterns are recommended by human experts. However, usually there are not enough process experts, especially in small organizations. Therefore, we need an AI expert to do it in an automatic way, where the biggest trouble lies in how to compute the matching degree between the project requirements and software process patterns in a universal way.

To tackle the above challenges, we propose a *Goal-Question-Metric* (GQM) based approach to recommending software process patterns. In this approach, we design scenario questions and answers for each software process patterns according to their description. When a new project comes, these questions are answered according to its specific requirements by the project manager (PM). By calculating the correlation between project requirements and all process patterns based on real answers and expected answers, a candidate list of software process patterns is recommended.

This paper makes the following contributions:

DOI reference number: 10.18293/SEKE2017-082

1) We present a GQM-based approach of software process pattern recommendation. Through questionnaires, we extract the structured scenario data from the textual description of software process patterns and project requirements. And then, the correlations between requirements and process patterns are calculated. As a result, software process patterns with high correlations are recommended.
2) We propose a topic clustering method to facilitate design scenario questions. We use LDA [6] to build a topic model for each sentences of software process patterns descriptions. After that, we calculate the similarities between sentences based on topic distribution, and use K-means [7] algorithm to do text clustering. With these clusters, we can design the scenario questions in a cluster-level instead of single sentence level, which can save lots of time and labor.

The remainder of this paper is organized as follows: Section II presents the related work; Section III describes our approach; Section IV discusses the experiments and results; Section V concludes the paper.

II. Related Work

Pattern is a common solution to a recurring problem in a given context [8]. Birukou et al. [9] divided the problem of reusing patterns into two steps, searching for patterns and selecting patterns. The problem of searching for patterns is to find appropriate patterns to solve given problems, and the problem of selecting patterns is to choose patterns to apply from a list of patterns. In the domain of software engineering, there are many different kind of patterns. Research mainly focus on the selection and recommendation of design patterns and process model.

A. Selection and Recommendation of Design Patterns

R. Mustapha et al. [10] proposed a recommender system for design patterns by labelling each patterns with several key words, and ask user to input some key words and about their project, then patterns are recommended according to key words matching. F. Palma et al. [11] proposed a recommendation method of software development design patterns. They extracted features manually from various design patterns, and then achieved scores of these extracted features by questionnaire. The matching degree between requirements and design patterns was calculated by the sum of answered weights. Sanyawong et al. [12] extracted names of classes and methods from software design. With these names, the similarity between different software designs can be calculated, and design patterns were recommended to novice designers. Issaoui et al. [13] used semantic information of class names, method names and description of software to do recommendation.

B. Selection and Recommendation of Process Model

In recent years, with the rapid development of the Internet and open source communities, lots of historical data about the software projects has been saved which can be used by researchers. Little et al. [14] proposed several attributes to score the suitability of the development process approach for a particular project. Egwali and Akwukwuma [15] proposed 35 criteria and a relative rating mechanism for these criteria to select appropriate process model. But these criteria rely heavily on the authors' subjective opinions which may affect the selection appropriate process models. Song et al. [16] defined a machine learning based method of software process model recommendation. Choosing an appropriate process model for a new project is achieved by utilizing the relationship between software project characteristics and the appropriate process models.

In summary, most of current research on patterns selection and recommendation mainly adopt features extracted from structured data. Kubo et al. [17] proposed a method to search appropriate unstructured patterns according to patterns' popularity, but this method can't match the requirement of a software project in detail.

III. Approach

A. Approach Overview

In order to deal with above problems, we propose a GQM-based approach to software process pattern recommendation, as shown in figure 1. The input is a reusable software process pattern library, where each pattern is described in the form of *Name*, *Intent*, *Domain*, *Solution*, *Initial Context* [4], as illustrated in Table I. The output is a recommended pattern list to a new project.

TABLE I: Description Example of Pattern FLOOT.

Item Name	**Content**
Name	Full Life Cycle Object-Oriented Testing, FLOOT
Intent	FLOOT methodology is a collection of testing techniques to verify and validate object-oriented software. The goal is to define software defects before delivery to users and assure your software applicable as a complete artifact.
Domain	Testing
Solution	1. Make testing plan for systematic testing; 2. Input systematic testing plan; 3. Commit systematic testing, including functional testing, pressure testing, installation testing and operation testing; 4. Record problem and defect for regression testing. Condition 1: If it passes regression testing, go to step 5; Condition 2: If it fails to pass regression testing, go to step 1. 5. After systematic testing, software is under user testing, including Alpha/Beta testing, delivery testing.
Initial Context	1. Your software is under package for delivery. FLOOT will test software as a whole for installation tools, documents and software. 2. Master test / quality assurance plans are finished. You need to manage and track testing work.

Our approach consists of two phases:

1) At questions and answer design phase, we preprocess scenario descriptions of process pattern, build topic model for each scenario sentence. After that, we do text

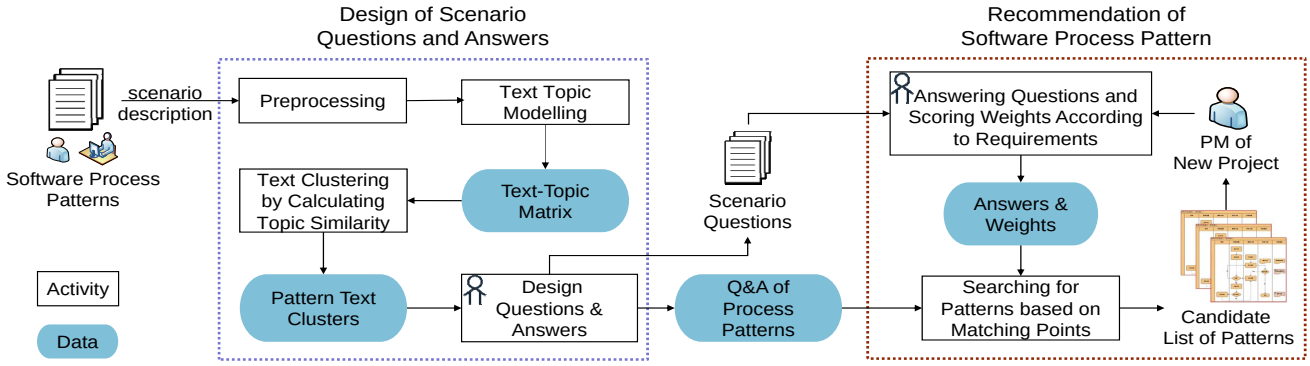

Fig. 1: Our Approach to Software Process Pattern Recommendation

clustering on all sentences for patterns by calculating similarity on text topic matrix. For each cluster, scenario questions are designed, and each question on every related pattern is assigned with an answer.

2) At pattern recommendation phase, PM firstly needs to answer related scenario questions and score the relevance between their requirements and questions. We get the candidate list of patterns recommended by calculating the matching degree between PM's answers and pattern answers designed in the previous phase.

B. Design of scenario questions and answers

Software process patterns are accumulated by experts' past experience and knowledge. However, such textual descriptions cannot be understood by machine. We adopt scenario questions to describe process pattern characteristics and new project requirements, for the further recommendation. Here, it will be described in details how the questions and answers are designed using the GQM model and machine learning technology.

1) Goal-Question-Metric Model: GQM [18] is a three layer model, where goals should be identified first, questions are designed to achieve these goals, and then questions are answered by metrics. In this paper, we apply GQM model to the questions design of process patterns. As Figure 2 shows, goals on the top layer are process patterns, questions on the middle layer are scenario questions, and metrics on the bottom layer are the answers to scenario questions, which are in the form of (Yes / No / Don't know) with a weight on a scale of 0-10. Each pattern has its scenario questions and corresponding answers.

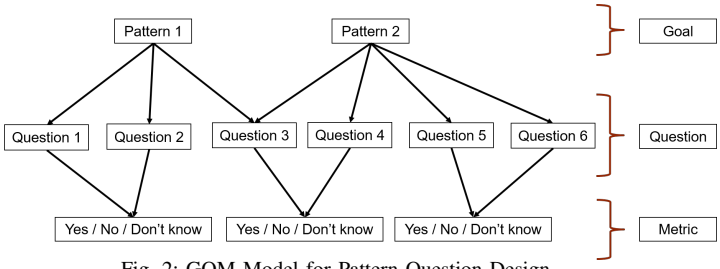

Fig. 2: GQM Model for Pattern Question Design.

2) Steps of Question Design: There are four steps in scenario question design.

- **Preprocessing**. The sentences in the *Intent*, *Domain* and *Initial Context* from pattern description (Table I) are extracted and filtered as the scenario descriptions of a pattern, and then represented in the form of bag of words.
- **Text topic modeling**. We apply LDA [6] on all the scenario sentences from all patterns to build topic model, and get text-topic probability distribution matrix. The size of matrix is d $\times$ k, where d means the amount of all sentences and k indicates the count of topics (in our method we set k with 100). Each row vector in matrix means topic probability distribution of one sentence, and the whole matrix represents topic probability distribution of all pattern descriptions.
- **Calculating topic similarity and sentences clustering**. We adopt K-means [7] to probability distribution matrix for sentences clustering. The similarity between texts is related to the similarity between vectors in matrix, which is calculated by Euclidean distance, as defined in formula (1).

$$distance(X,Y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \qquad (1)$$

where X,Y is the row vector which consists of x_i and y_i; x_i and y_i are respective probability of the sentence belonging to the i^{th} topic; and k is the number of all topics. After text clustering to all the pattern description sentences, we can obtain the topic cluster collection of all the sentences. In our method, we set K for K-means with 25.

- **Design of questions and answers**. We design scenario questions according to all clusters of pattern descriptions. Each pattern has one or more questions. As the example in Figure 3 shows, each pattern description consists of several sentences. After topic clustering, each cluster contains various sentences from different patterns. Questions are mainly from the clusters sentences. Once all questions have been designed for all clusters, we assign a right answer to each question on every related pattern.

3) An Example: Here is an example of how to design scenario questions and answers. Table II shows the scenario

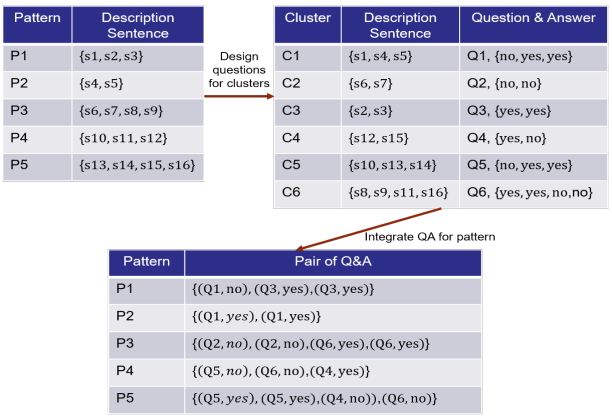

Fig. 3: Patterns, Clusters and Questions.

descriptions of three patterns, waterfall development (WD), iterative development (ID), and prototype development (PD). Table III lists the topic clustering results and their designed questions. And Table IV shows the scenario questions and answers for each pattern, where '—' means the question is not related to the pattern.

TABLE II: EXAMPLE OF SCENARIO DESCRIPTION.

Pattern Name	Pattern Description Sentences
Waterfall development	1. Be useful for complex system whose requirements should be stable and clear. 2. Be suitable for the projects with less risks. 3. Places emphasis on documentation. 4. Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
Iterative development	5. The requirements could be not so stable at the beginning. 6. Offer many different available versions. 7. Often contact with users to obtain feedback. 8. Can reach a high demand of risk control.
Prototype development	9. Be suitable for small project whose requirements are not very clear. 10. Work with users during the development. 11. Ensure the developed software with a high usability. 12. May delay the whole development cycle.

TABLE III: TOPIC CLUSTERING AND QUESTION RAISE.

Sentence Clusters	Scenario Questions
{1,5,9}	1. Are the requirements of your project stable and clear?
{2,8}	2. Does your project have some high risks need to control?
{3}	3. Do you need detailed documents?
{4,12}	4. Do you need a complete development and strict plan?
{6}	5. Do you need to release system versions quickly?
{7,10}	6. Do you need to contact with users often during development process?
{11}	7. Does your project need a high usability?

C. Recommendation of Software Process Pattern

In this phase, appropriate patterns will be recommended according to the requirement of a new project. Our approach

TABLE IV: EXAMPLE OF Q&A DESIGN FOR PATTERNS.

Question	WD	ID	PD
1. Are the requirements of your project stable and clear?	Yes	Yes or No	No
2. Does your project have some high risks need to control?	No	Yes	—
3. Do you need detailed documents?	Yes	—	—
4. Do you need a complete and strict development plan?	Yes	—	No
5. Do you need to release system versions quickly?	—	Yes	—
6. Do you need to contact with users often during development process?	—	Yes	Yes
7. Does your project need a high usability?	—	—	Yes

obtains project requirements by questionnaire and uses scenario questions to match process patterns. The benefit of questionnaire is that it avoids analyzing the textual requirements of a new project by nature language processing technology, and helps understand them more accurately.

1) Recommendation Method: Figure 4 shows the recommendation method of software process pattern. When a new project coming, PM will answersthe scenario questions with "Yes/No/Don't know" and score the weights between the project requirements and questions. The weight is an integer from 0 to 10 for quantification of the relationship between patterns and project requirements. After recording the answers and weights, we can calculate all the matching degrees between patterns and questions. As a result, we sort all the matching degrees and pick the first five process patterns as the candidate list to PM. In addition, it is not all questions that should be answered. Questions are often correlative, so we will filter the remaining questions based on the answers of previous questions.

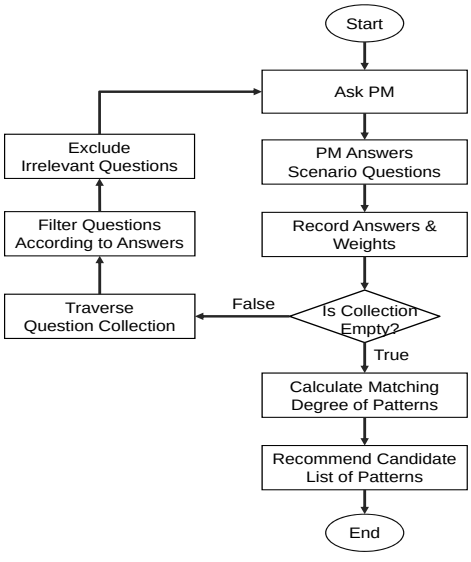

Fig. 4: Recommendation Method

To recommend the appropriate patterns for a project, the matching degree between their answers of scenario questions is calculated by formula (2).

$$MD = \frac{1}{NQ}(\sum_{i=1}^{n}(Weight_{i\_same)} - \sum_{j=1}^{m}(Weight_{j\_diff)}) \quad (2)$$

In the formula, MD is the matching degree between patterns and new project requirements; NQ is the number of all related questions for the pattern; $\sum_{i=1}^{n}(Weight_{i\_same)}$ is the weight sum of all PM's answers in accordance with pattern's answers; $\sum_{j=1}^{m}(Weight_{j\_diff)}$ is the weight sum of all PM's answers inconsistent with pattern's answers. For example, there is a question collection {Q_1,Q_2,Q_3} and their pattern answers are {yes,no,no}. A PM answers these questions and gives {(yes,10),(yes,8),(no,9)}. The questions with the same answers are Q_1 and Q_3, while Q_2 is the question with different answer. As a result, $\sum_{i=1}^{n}(Weight_{i\_same\_answer)}$ is 19 and $\sum_{j=1}^{m}(Weight_{j\_diff\_answer)}$ is 8. So the matching degree between this pattern and PM's requirements is $\frac{11}{3}$.

2) An Example: Here is a small example to illustrate which pattern is prefered. Company A needs to develop a graphic software. The requirement is not very clear and the client ask for a high usability. In order to get appropriate software process patterns, PM needs to answer the scenario questions. Table V shows the records of pattern recommendation questionnaire. In the table, column 'A' represents the answer to question and 'W' is the weight. Three patterns are examined, including waterfall development pattern (WD), iterative development pattern (ID), and prototype development pattern (PD).

Take waterfall development pattern as an example, its related question collection is {Q_1,Q_2,Q_3,Q_4} and corresponding answers are {yes,no,yes,yes}. PM's answer pairs are {(no,8),(yes,5),(no,5),(yes,3)}. Only Q_4 is matched. So Weight$_{same_answer}$ is 3, Weight$_{diff_answer}$ is 18 and the matching degree between pattern waterfall development and PM's requirements is -$\frac{15}{4}$. According to the total weight, prototype development pattern is the optimal fit among these three patterns.

TABLE V: RECORDS OF PATTERN RECOMMENDATION QUESTIONNAIRE.

Question	**A**	**W**	**WD**	**ID**	**PD**
1. Are the requirements of your project stable and clear?	No	8	-8	8	8
2. Does your project have some high risks need to control?	Yes	5	-5	5	—
3. Do you need detailed documents?	No	5	-5	—	—
4. Do you need a complete and strict development plan?	Yes	3	3	—	-3
5. Do you need to release system versions quickly?	No	8	—	-8	—
6. Do you need to contact with users often during development process?	Yes	8	—	8	8
7. Does your project need a high usability?	Yes	10	—	—	10
Total Weight	—	—	-$\frac{15}{4}$	$\frac{13}{4}$	$\frac{23}{4}$

TABLE VI: PERFORMANCE COMPARISONS OF TEXT CLUSTERING.

Clusters	Num Of Sentences	TF-IDF			LDA		
		P	R	F	P	R	F
Architecture	11	0.63	0.64	0.63	0.72	0.73	0.72
Documentation	8	0.83	0.63	0.71	0.75	0.75	0.75
Design	25	0.65	0.60	0.62	0.82	0.76	0.79
Governance	19	0.71	0.53	0.61	0.72	0.68	0.70
Testing	12	0.50	0.75	0.60	0.75	0.83	0.79

IV. EXPERIMENTS

In this section, we conduct experiments to answer these two research questions:

RQ1: Compared to traditional text clustering approaches based on statistics, does our approach reach a better performance?

RQ2: Does our approach recommend appropriate software process patterns to a new project?

A. Experimental Settings

Ambler published an online software process pattern collection*. We extract totally 378 description sentences for 89 patterns from this website. For RQ1, we use Precision, Recall and F1-Measure as the evaluation criteria. For RQ2, we use HitRate to calculate the precision of pattern recommendation, as defined in formula (3), where *hitCnt* means the number of successful recommendation and *TotalCnt* means the total number of recommendation.

$$HitRate = \frac{hitCnt}{TotalCnt} \quad (3)$$

B. Text Clustering Experiment

To answer RQ1, we conduct a comparison experiment between our approach and the statistics approach based on TF-IDF. We select 75 sentences from all these 378 description sentences and divide them into 5 topics manually, as the benchmark. At the same time, the automatic text clustering is made by two approaches, where the topic number for LDA is set with 100 and the target clusters number of K-means with 5.

The experimental result is illustrated in Table VI. It shows that our approach based on LDA has higher precision and recall compared traditional TF-IDF approach, which has better effect on text clustering of software process pattern descriptions.

C. Pattern Recommendation Experiment

To answer RQ2, we collect 30 software projects which have been closed successfully and record their corresponded process patterns manually as the benchmark. We invite 18 PMs with different professional level to involve in this experiment. For each PM, we select 5 projects randomly and offer them enough time to understand the requirements of selected projects. Then they follow our approach to obtain recommended patterns.

*http://www.ambysoft.com/processPatternsPage.html

In a recommendation, if one of the top five recommended patterns matches with the recorded patterns, we regard this recommendation as a successful one.

In our method, Totally 57 questions are designed for all these 89 patterns we extract. Table VII (HR denotes hit recommendations and TR denotes total recommendations) lists the recommendation results for different PM professional levels. It shows that the average precision of pattern recommendation can reach 57%, which can meet the managers' requirements on recommending software process patterns.

TABLE VII: RECOMMENDATION RESULTS.

Professional Level of PMs	Num of PMs	HR	TR	HitRate
Beginner	4	10	20	50%
Normal	4	11	20	55%
Medium	4	13	20	65%
Advanced	4	12	20	60%
Skilled	2	6	10	60%

D. Comparison with other methods

Considering other two state-of-the-art methods mentioned in [14] and [16]. For the first method, it only uses 4 kinds of different process model, so it is not suitable to do a direct comparison. For the second method, a huge amount of historic software project data are collected to employ this model., it is hard for us to obtain satisfied data for redoing their experiments. So we just analyze the pros and cons of different methods as shown in Table VIII. Our approach can deal with unstructured textual descriptions of software patterns, and thus is more practical and can be applied in more scenarios.

TABLE VIII: COMPARISON OF DIFFERENT APPROACHES.

Method	Pros	Cons
Our Method	accurate recommendation; expandable pattern repository; unstructured data understandable	semi-automatic questions design; Rely on the answering of the PM, not so objective
[14]	Raise criteria to measure complexity and uncertainty; Divide projects into 4 categories according to complexity and uncertainty	unstable precision; divide all process patterns into 4 category, not specific enough
[16]	quantified software development attribute; accurate recommendation	inapplicable to unstructured data(such as requirements); not convenient to add new process patterns

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a GQM based software process recommendation approach. The approach applies GQM model to design scenario questions for different patterns, with the help of natural language processing and machine learning techniques. For a new project, its requirements are elicited through answering these questions, and then the optimal matching patterns are recommended. Experimental results show our approach can recommend the proper process patterns to a specific project with a high precision.

As for future work, we will explore an automatic approach to generate scenario questions from the descriptions of software process patterns. And we also plan to do more experiments to compare our approach with others, apply our approach in practice and improve it according to the feedback.

ACKNOWLEDGEMENT

Beijun Shen is the corresponding author. This research is supported by National Natural Science Foundation of China (Grant No. 61472242) and 973 Program in China (Grant No. 2015CB352203).

REFERENCES

[1] W. W. Royce *et al.*, "Managing the development of large software systems," in *proceedings of IEEE WESCON*, vol. 26, pp. 1–9, Los Angeles, 1970.

[2] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[3] K. Beck, "Embracing change with extreme programming," *Computer*, vol. 32, no. 10, pp. 70–77, 1999.

[4] S. W. Ambler, *Process patterns: building large-scale systems using object technology*. Cambridge University Press, 1998.

[5] S. W. Ambler, *More process patterns: delivering large-scale systems using object technology*. Cambridge University Press, 1999.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[7] J. B. Macqueen, "On convergence of k-means and partitions with minimum average variance," *Annals of Mathematical Statistics*, vol. 36, 1965.

[8] J. M. Küster, C. Gerth, A. Förster, and G. Engels, "Detecting and resolving process model differences in the absence of a change log," in *International Conference on Business Process Management*, pp. 244–260, Springer, 2008.

[9] A. Birukou, "A survey of existing approaches for pattern search and selection," in *Proceedings of the 15th European Conference on Pattern Languages of Programs*, p. 2, ACM, 2010.

[10] Y. G. Guéhéneuc and R. Mustapha, "A simple recommender system for design patterns," *Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories*, 2007.

[11] F. Palma, H. Farzin, Y. G. Guéhéneuc, and N. Moha, "Recommendation system for design patterns in software development: An dpr overview," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, pp. 1–5, IEEE Press, 2012.

[12] N. Sanyawong and E. Nantajeewarawat, "Design pattern recommendation based-on a pattern usage hierarchy," in *International Computer Science and Engineering Conference (ICSEC)*, pp. 134–139, IEEE, 2014.

[13] I. Issaoui, N. Bouassida, and H. Ben-Abdallah, "A new approach for interactive design pattern recommendation," *Lecture Notes on Software Engineering*, vol. 3, no. 3, p. 173, 2015.

[14] T. Little, "Context-adaptive agility: Managing complexity and uncertainty," *IEEE Software*, vol. 22, no. 3, pp. 28–35, 2005.

[15] A. O. Egwali and V. V. N. Akwukwuma, "Security framework for software process models: Measures for establishing a choice," *Asian Journal of Information Technology*, no. 1, pp. 463–471, 2012.

[16] Q. Song, X. Zhu, G. Wang, H. Sun, H. Jiang, C. Xue, B. Xu, and W. Song, "A machine learning based software process model recommendation method," *Journal of Systems and Software*, vol. 118, pp. 85–100, 2016.

[17] A. Kubo, H. Nakayama, H. Washizaki, and Y. Fukazawa, "Patternrank: A software-pattern search system based on mutual reference importance," *15th Pattern Languages of Programming (PLoP)*, 2008.

[18] N. E. Fenton, *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, 1996.